CPSC 359 – Fall 2019
Assignment 3: Basic GPIO and Interrupts

based on L. Manzara 2019

due: 0900h 18-Nov-2019

Background

This assignment was originally created by Dr. Leonard Manzara for CPSC 359, Winter 2019.

In this assignment you will begin to use the memory-mapped input/output (I/O) features of the BCM2837 *system on chip* that is at the heart of the Raspberry Pi. You will also use interrupts so that your program responds efficiently to changes in inputs.

Objective

Your goal is to write a program, in *C*, that illuminates three LEDs in a given pattern (order and duration). The LED pattern should change, as specified, in response to changes in the input.

Table 1 lists the Raspberry Pi I/O pins that you will use. These are the pins corresponding to LEDs 1, 2, and 3, and inputs 1 and 2 on the ExplorerHat Pro boards in the lab. Note that all of the ExplorerHat inputs are pulled low.

GPIO pin	Input or Output	Device
4	output	LED 1
17	output	LED 2
27	output	LED 3
23	input	Input 1 (pulled low)
22	input	Input 2 (pulled low)

Table 1: GPIO pin usage for assignment 3.

There are two LED patterns that your program will produce. These are:

1. The LEDs illuminate sequentially in the order: 1, 2, 3, 1, 2, 3, and so on. Each will be on for $0.5s$, then off for a delay of $0.5s$.
2. The LEDs illuminate sequentially in the order : 3, 2, 1, 3, 2, 1, and so on. Each will be on for $0.25s$, then off for a delay of $0.25s$. I.e., the reverse order of the first pattern, and faster.

When your program starts, it should illuminate the LEDs in the first pattern. Upon receiving an interrupt from *Switch A*, the LEDs should change to the second illumination pattern. Similarly, when your program receives an interrupt from *Switch B* the LEDs should change to the first pattern. Although it would be possible to do this by polling, your program must use interrupts on the the two inputs to achieve the specified behaviour.

Your program must be written in the *C* language, with the exception of the provided assembly code that initializes the Pi and sets up the IRQ vector.

Details/Hints

You may use code from examples provided in the course material through D2L – but be sure to note in your source code comments where the code came from.

You should use the system timer counter register to time your delays. See the Broadcom ARM Peripherals manual, p 172. It is not more efficient than than using delay loops, but it does expose you to another, easy-to-use capability of the Raspberry Pi.

As stated above, use the *C* language because: it will give you more practice with an important programming language, and it is much easier to work with then assembly language. In fact, this kind of low-level machine programming is precisely what *C* does best. See `https://en.wikipedia.org/wiki/C_(programming_language).`

As with any programming project, I recommend that you approach this incrementally. Start by implementing a simple subset of the required functionality, then add the remaining functionality one piece at a time. I suggest doing the interrupts last.

Debugging is hard to do in this bare metal environment. Doing incremental changes will help you to zero on the source of errors. Also, UART output may help to find some bugs, so set that up.

Good coding practices include all the good habits you should exhibit when writing your code. These include, but are not limited to:

- indentation,
- informative variable and function names,
- helpful comments,
- the presence additional, helpful files such as readme.txt and Makefile, and
- code modularity.

Deliverables

Turn in the following items for evaluation.

1. All files required to build your program. This includes all of your source code files, Makefile, and a readme.txt file with instructions.
2. A demo of your working program for your TA during a tutorial session – date to be determined by your TA.

Bonus (10%)

Use the system timer peripheral or the timer (ARM side) to trigger interrupts, to create any delays needed in your program. In other words, replace your *busy loop* with code that uses one of the hardware timers to delay the appropriate amount of time.

Note that this is not the same as implementing a delay loop by polling the system timer. You must specifically have interrupts triggered by a timer.

Evaluation

1	Program runs as specified	
	LED Pattern 1	3pts
	LED Pattern 2	3pts
	Response to buttons	4pts
2	Configuration of I/O pins	4pts
3	Configuration of interrupts	3pts
4	Coding practices	3pts
5	Timer interrupts (bonus)	2pts
	Total	20pts

There is no team work. Each students must submit individual solutions.

Late work

After the deadline and up to 24hrs late: -5pts. After 24hrs and up to 48hrs late: -10pts. Over 48hrs late: -20pts, i.e., no assignment will be accepted beyond 48hrs after the deadline.

Plagiarism

Work must me the sole work of the individual submitting the work.

You may find that the this assignment is similar to other published code. Nevertheless, you should go through the process yourself, and submit your own work. If you do borrow from other sources, cite the source and clearly indicate what you have borrowed, keeping in mind the design must be substantially your own. If you cite your sources, worst case you may receive a reduced grade for borrowing too much. If you borrow, but do not cite, that is plagiarism and academic misconduct, and carries severe penalties as determined by the Faculty of Science. You may use code provided in the course content on D2L, but you must still cite the source.

As a guideline, consider the 20-minute rule. Talk with your colleagues and consult other sources (cite them please). Wait at least 20 minutes, then do your work to be sure that it is your own. Less then 20 minutes usually means that you are merely copying work from the original source.